



Universitat d'Alacant
Universidad de Alicante

Grado en ingeniería robótica Robots móviles

Trabajo teórico:
Robot SPOT - Boston Dynamics



Hecho por:
Nuria González Hernani
Elvira María Rodríguez Baídez

ÍNDICE

Introducción	3
Historia de Boston Dynamics	3
Evolucion Robots cuadrupedos	3
Robot Spot - Boston Dynamics	5
Aplicaciones	5
Hardware	6
Capacidades Añadidas	7
Software	10
Tablet	12
Scout	13
Spot SDK	14
Simulación	18
Enlace al drive	22
Bibliografía	22

Introducción

Historia de Boston Dynamics

Boston Dynamics es una empresa de ingeniería y robótica fundada en el año 1992 como parte del instituto tecnológico de Massachusetts, Estados Unidos. Es conocida por el desarrollo de distintas series de robots que destacan por sus movimientos dinámicos.

En sus inicios, Boston Dynamics trabajó con American Systems Corporation con el objetivo de ayudar a la División de Sistemas de Entrenamiento del Centros de Guerra Aérea Naval a reemplazar los videos usados en entrenamiento por simulaciones interactivas en 3D.

Más adelante la compañía empezó con la creación de robots físicos. Su objetivo era ser capaz de fabricar y programar robots con la habilidad de realizar movimientos parecidos a los animales aunando agilidad y destreza con la capacidad de percepción e inteligencia.

Aunque en un principio esta empresa solo hacía labores con fines de investigación y desarrollo, en 2019 el robot Spot empezó a comercializarse.

Evolucion Robots cuadrupedos

BigDog fue el primer robot creado por Boston Dynamics en el año 2004. Éste era capaz de circular por terrenos difíciles usando sensores y un sistema de visión.



Imagen 1: Robot BigDog

Durante 2010 se desarrolló el robot LS3 con el fin de mejorar el funcionamiento de BigDog y usarlo con fines militares, soportando así mayores cambios en temperatura y ambientes más sucios. Además, era capaz de responder a comandos orales y visuales junto con geolocalización y sensores LIDAR.



Imagen 2: Robot LS3

Por otro lado, en 2012 se desarrolló WildCat, versión mejorada de Cheetah, cuya característica más destacable es la capacidad de correr a más de 25 km/h de manera autónoma y manteniendo el equilibrio por sí mismo usando bombas hidráulicas.



Imagen 3: Robot WildCat

Finalmente, en 2015 se fabricó el Spot Classic, alimentado eléctricamente y accionado de forma hidráulica capaz de moverse por terrenos difíciles y realizar movimientos como subir escaleras. Es el antecesor del Spot desarrollado en 2020, del que se hablará más en profundidad en los siguientes puntos.

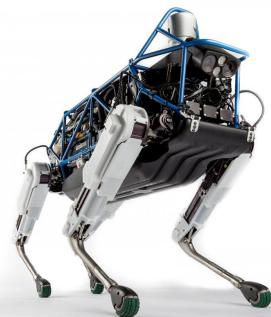


Imagen 4: Robot Spot Classic

Robot Spot - Boston Dynamics

Spot es un robot móvil con cuatro patas que destaca por su gran agilidad y capacidades de movilidad, así como el hecho de que está diseñado como una plataforma escalable que le permite evolucionar constantemente para poder realizar mayor variedad de trabajos aburridos, desagradables o peligrosos. Sus múltiples sensores, además, le permiten recopilar grandes cantidades de información que puede utilizarse para crear modelos de mantenimiento predictivo, construcción de copias digitales (digital twins) o minimizar el riesgo de los empleados.

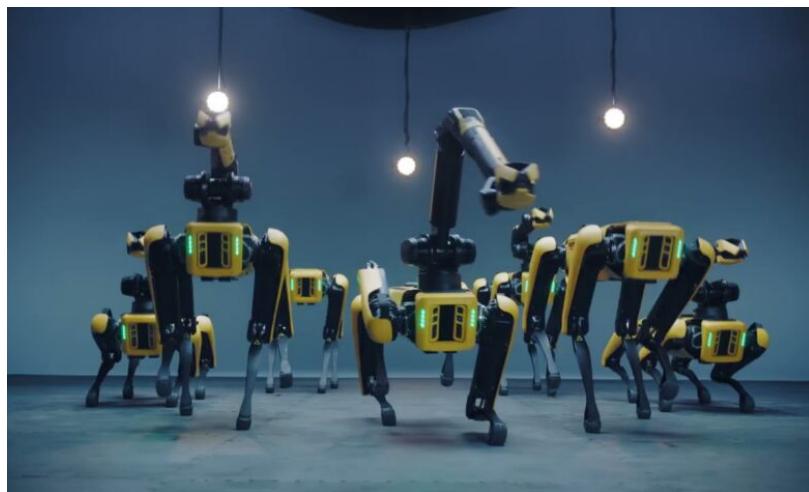


Imagen 5: Robots de Boston Dynamics

Aplicaciones

Su función, por lo tanto, es moverse por diferentes terrenos, analizar sus alrededores con ayuda de diferentes sensores y equipamiento y, si es necesario, actuar en el entorno.

Todas estas características son idóneas para llevar a cabo diferentes aplicaciones en multitud de industrias como son la manufacturación, construcción, energía, investigación, minería, aceite y gas, seguridad pública e incluso entretenimiento.

Algunas de las aplicaciones que se pueden llevar a cabo en estas industrias son únicas, pero otras muchas son comunes. Estas últimas incluyen actividades como: realizar tareas autónomas de inspección (para detectar fugas de aire, hacer lecturas de válvulas, detectar radiación...), crear copias digitales (digital twins) de la planta, supervisión del progreso, desarrollo de aplicaciones, monitorización de zonas peligrosas o poco accesibles para el ser humano, etc.

Hardware

En términos de hardware, SPOT es un robot de dimensiones 1.1x0.5 metros, una altura aproximada de 0.61 m y un peso de casi 33 kg. Puede alcanzar una velocidad de 1.6 m/s y está equipado con una serie de sensores y actuadores que le permiten moverse e interactuar con su entorno.

Estos incluyen cuatro patas articuladas con motores y sensores de par, una cabeza montada con cámaras de visión y láseres para la navegación y la evasión de obstáculos, y una serie de sensores en sus patas y cuerpo para la percepción del terreno y el balance. También cuenta con una batería recargable y un sistema de comunicación inalámbrico para la conexión con otros dispositivos y la transmisión de datos.

Spot tiene dos actuadores en cada cadera (HX, HY), con 45 y 91° de rotación respectivamente, y uno en cada rodilla (Knee), con una flexión/extensión de 14 a 160°. Todo ello le otorga al robot 12 GDL, tres por cada pierna. Cada pierna, además, se clasifica en izquierda, derecha, delante (front) y atrás (hind). Por tanto, para referirse a una articulación concreta de las piernas del Spot, se usan acrónimos como FL.HY (front left hip Y) para decir la pierna izquierda de delante, articulación de la cadera eje Y.

Por otro lado, su estructura le permite cargar con hasta 14 kg de equipamiento de inspección. Con ayuda del láser, que le permite visualizar 360° y una distancia máxima de 4 metros, puede crear un mapa del terreno para evitar obstáculos de forma local (según aparecen a su alrededor). Además, una unidad de medida inercial (IMU) y cuatro sensores de fuerza/posición (una en cada pata) le ayudan a mantener un equilibrio dinámico para moverse por entornos irregulares (tierra, césped, escaleras, bordillos...).

Además del equilibrio, SPOT usa un sistema de percepción para evitar de forma automática la colisión con obstáculos. Este sistema de percepción consiste en cinco cámaras estéreo: dos al frente, una a cada lado y una atrás del robot. Estas cámaras devuelven información de profundidad e imágenes a color. El robot no necesita estas imágenes para funcionar, pero ayudan a mejorar la visión del operador para que comprenda de manera más intuitiva lo que ocurre alrededor del robot.

Por otro lado, tiene conexión Wifi y Ethernet, 2 puertos de conexión y una batería con una autonomía de 90 minutos.

Por último, con el objetivo de llevar a cabo tareas muy variadas y personalizadas, el robot está adaptado para poder conectar e integrar módulos externos de hardware (brazo robótico, unidad de iluminación, sensores...) mediante rieles de montaje y puertos de carga útil (payload ports).

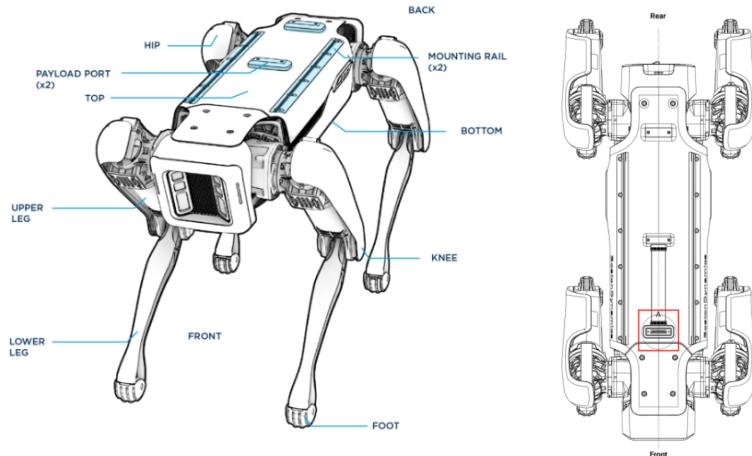


Imagen 6: Especificaciones del Spot

Capacidades Añadidas

Percepción: Para mejorar la percepción, se puede equipar al Spot con una cámara esférica de 360 x 170° (Spot CAM+ y Spot CAM+IR), que permite las tareas de vigilancia o emisión de video en stream, además de incluir altavoces y micrófonos. Los FPS de las imágenes percibidas dependen de la señal del wifi, sin embargo la resolución de la cámara es de 2MP y 1080p con una capacidad de zoom de 30x.



Imagen 7: Robot Spot CAM+ y Spot CAM+IR

Computación: Para mejorar la computación, se le puede añadir en la parte superior del robot una CPU externa dedicada (Spot CORE y Spot CORE I/O), que servirá para probar códigos creados por desarrolladores en el propio robot, con la posibilidad de utilizar los códigos nuevos mediante conexión de Internet. Esta CPU cuenta con un procesador i5 Intel de 8th Gen, 512 GB SSD y 16 GB DDR4. Como sistema operativo se utiliza Ubuntu Desktop 18.04 LTS.

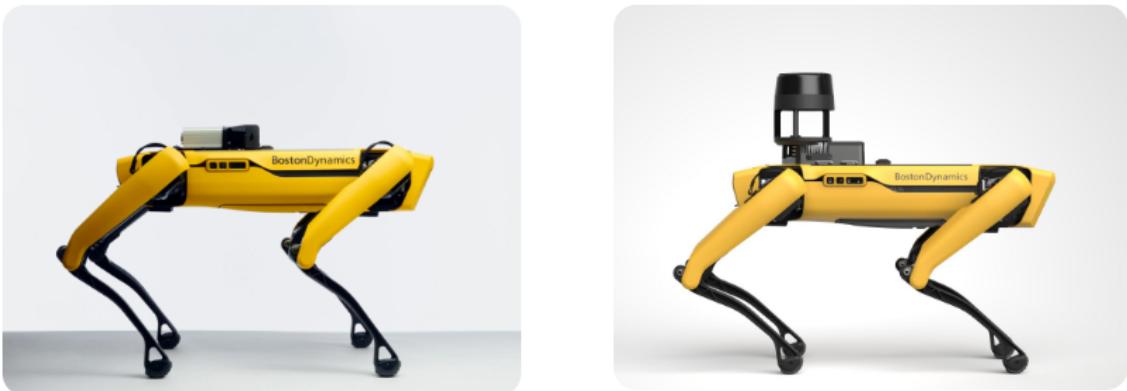


Imagen 8: Robot Spot CORE y Spot CORE I/O

Autonomía: Si se quiere mejorar la creación de mapas, siendo estos más precisos y de un mayor tamaño, se puede equipar al Spot con un sensor LIDAR (Spot EAP), mejorando el rango de las cámaras 3D que dispone el robot en su base. Este sensor modelo Velodyne VLP-16 permite realizar medidas a 100 metros de distancia con una precisión de +- 3cm.



Imagen 9: Robot Spot EAP

Integración: Si se quieren añadir nuevos puertos, contamos con el Spot GXP, que añade un puerto de Ethernet, HD15 y un puerto para carga de 5/12/24V que se puede regular.



Imagen 10: Robot Spot GXP

Comunicación: Para mejorar la comunicación, se puede equipar al Spot tanto con un Kit de radio (Rajant Kinetic Mesh Radio Kit) como con nuevos puertos de entrada y salida (Spot CORE I/O). El kit de radio permite establecer una conexión robusta para teleoperar al robot en vez de hacerlo mediante una WIFI externa. Además, contamos con seguridad para encriptar tanto los mensajes como las direcciones Mac.



Imagen 11: Robot Rajant Kinetic Mesh Radio Kit y Spot CORE I/O

Manipulación: Con el equipamiento de un brazo (Spot Arm), el Spot es capaz de realizar tareas de pick and place, incluyendo objetos como válvulas, herramientas o pomos de puertas. Esto permite que el robot pueda abrir o coger objetos realizando movimientos semi-automáticos preprogramados. Con ayuda de la Api, se puede realizar tanto un control en el espacio articular definiendo el ángulo de las articulaciones y las velocidades de éstas, como un control en el efecto final especificando la posición, velocidad y fuerza en espacio cartesiano. Este manipulador cuenta con 6 grados de libertad y es capaz de elevar una carga de 11 kg con una fuerza máxima de 130 N. También es resistente al agua y al polvo.

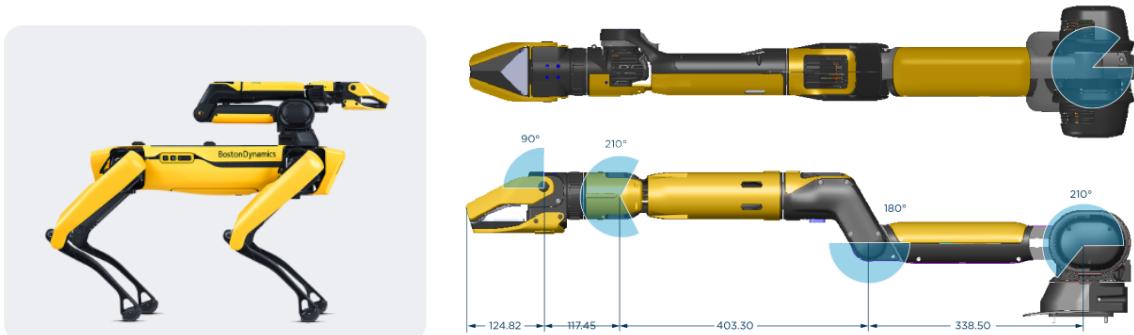


Imagen 12: Robot Spot Arm

Software

Para Spot, el mundo está representado como un grafo local con bordes y puntos (waypoints).

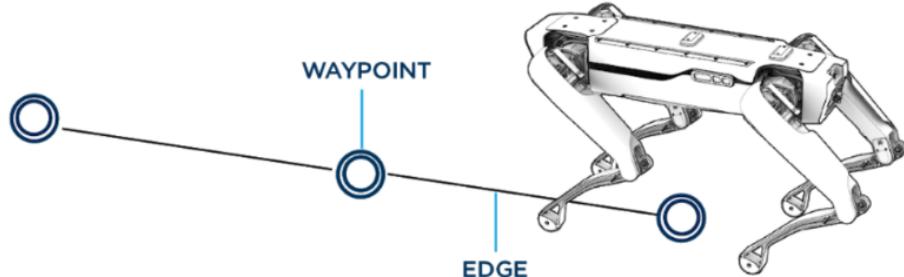


Imagen 13: Mapa de grafos

Los waypoints representan lugares en el mundo a los que el robot puede navegar. Cada waypoint tiene información asociada (snapshot) que recopila y analiza el robot con sus sensores desde su posición. Estos datos permiten al robot rastrear su posición a medida que se mueve de un waypoint a otro. Las zonas vacías y sin mucha textura, por tanto, no proporcionan buenas características para que el robot realice un seguimiento de su ubicación. Para solucionarlo, se pueden utilizar marcas como códigos QR en estos entornos. Otra alternativa son las cámaras LIDAR, que obtienen muchas más información de cada waypoint.

Los bordes representan cómo se mueve el robot entre puntos y contienen tanto la pose relativa de los puntos conectados como cualquier parámetro necesario que describa cómo debe moverse el robot.

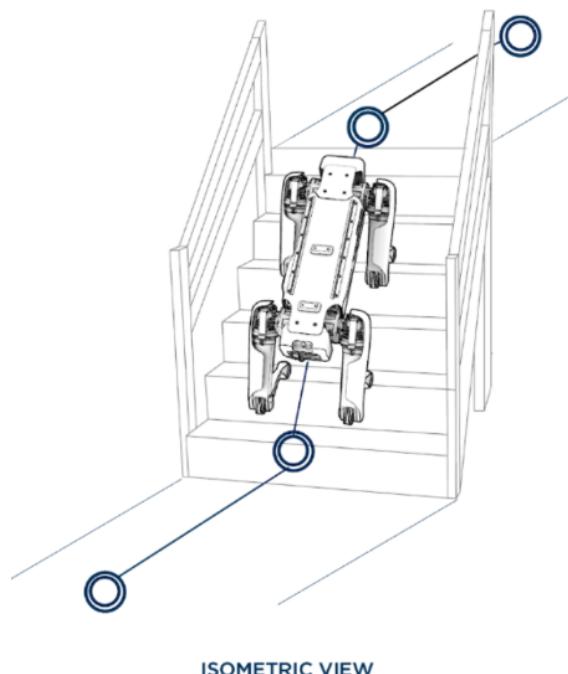


Imagen 14: Spot trasladándose de un punto a otro

Para mapear y guardar el entorno es necesario controlar el robot de forma manual. El grafo que resulta de este mapa se puede usar para localizar al robot. Una vez conectado al grafo del mapa, puede moverse a cualquier punto recorriendo el grafo de un punto a otro. Cabe destacar que los mapas no se guardan si se reinicia el robot (reboot). Para no perderlo, por tanto, es recomendable descargar el mapa del Spot, por ejemplo, usando la tablet.

Una vez se carga un nuevo mapa, el robot debe localizarse en el mapa. Una forma de hacerlo es dejar que el operario indique dónde se encuentra el robot en el mapa, para que el robot guarde la distancia a la que está del waypoint más cercano, y otra es iniciar el robot cerca de una marca distintiva como un QR que se haya registrado en el mapa durante su creación.

Usando la localización inicial, el robot analizará su posición relativa a otros waypoints mientras se desplaza para terminar de localizarse. Cabe destacar que no existe una posición dentro de un mapa global, es decir, que, cuando se mueve, el robot va cambiando su localización de un punto a otro del grafo, pero el mapa no se actualiza cuando esto ocurre, sino que se usa tal cual se creó como referencia para la navegación.

Actualmente, el robot no es compatible con dispositivos de localización externos como el GPS, por lo que es más probable que se pierda al navegar; principalmente debido a entornos con pocas características o con características que cambian mucho de repente. Para recuperarse de una condición de pérdida, la localización del robot debe reiniciarse. Si el problema persiste, los operarios pueden actualizar el mapa registrando nuevos puntos de referencia con nuevos datos.

En cualquier caso, por motivos de seguridad, mientras el robot esté en marcha, las personas deben estar en todo momento a un mínimo de 2 metros del robot.

Se puede controlar el robot de forma remota mediante un software de Spot, llamado Scout, o con una aplicación de tablet intuitiva que usa unas cámaras estéreo del robot para que el operario visualice el entorno del robot.

Tablet

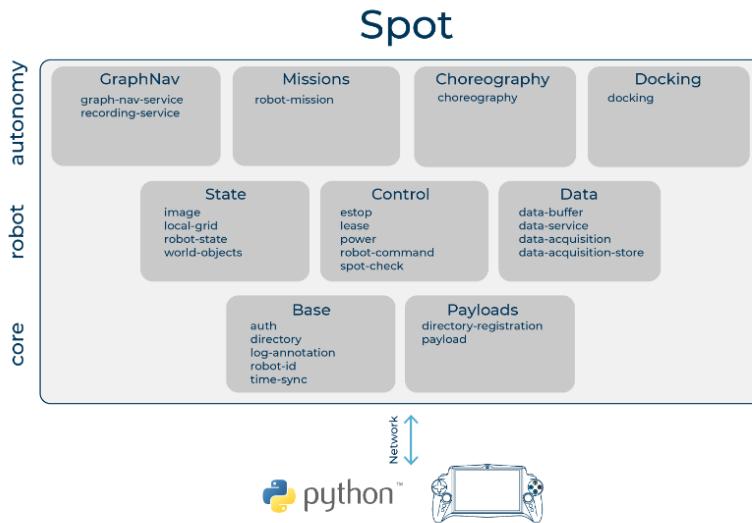


Imagen 15: Bloques Software del Spot

Estas cámaras se utilizan como sistema de percepción para crear un mapa en 3D la evitación de obstáculos automática. Por defecto, Spot está programado para mantener una distancia mínima de 0.075 metros con objetos cercanos. Sin embargo, los operadores pueden modificar esta distancia hasta 0.5 metros.

Por otro lado, con ayuda de la tablet se pueden tanto teledirigir al robot como programar misiones autónomas que se vayan a repetir en el tiempo y recopilar datos consistentes para analizar. Este software ha pasado por muchas mejoras y optimizaciones para asegurar que el control sea fácil de usar por el operario, pero suficientemente robusto para alcanzar las expectativas del usuario.

Por estas razones, esta tableta incluye: una pantalla táctil intuitiva y dos joysticks para mover a Spot y guardar misiones, más espacio en la pantalla para editar las misiones, teleoperación, mejor visualización, menor peso, protección a prueba de agua y una batería de 8 horas. Además, en caso del modelo Spot Arm, que incluye un brazo robótico, se cuenta con un controlador adicional que incluye un joystick, dos botones de cinco direcciones cada uno y dos botones bumper. Todo ello para asegurar la máxima capacidad de manipulación durante la conducción del robot.

Por ejemplo, los operadores pueden utilizar una función llamada Autowalk, que se basa en GraphNav (mostrado en la imagen superior) y en el servicio de Misión, para registrar y reproducir rápidamente recorridos en un entorno a la vez que se realizan acciones útiles por el camino. Autowalk también permite descargar los mapas guardados en los Spots a la tablet (para no perderlos en caso de reiniciarlo), así como cargar mapas en los robots.

Otra función de vital importancia es la parada de seguridad (STOP), que ofrece una forma segura de apagar el motor del Spot y es prioritaria a cualquier otra acción o control que se esté haciendo sobre el robot. Cuando STOP se ejecuta, el motor se detiene y el robot colapsa en el suelo. Se trata, por tanto, de una parada de emergencia, donde los

motores no se podrán iniciar hasta que STOP sea cancelado. El Spot también se puede parar automáticamente en casos de pérdida de señal (tras 3 segundos Spot se sienta, tras 8 segundos se apagan los motores) o en caso de caídas (utiliza las piernas para minimizar el daño de la caída).

Existen muchas otras funcionalidades y explicaciones de cómo se puede controlar al Spot con la tablet, sin embargo, se limitará a mencionar que muchas funciones del movimiento del robot se pueden realizar tanto con los joystick como con los botones siguiendo las indicaciones de la pantalla interactiva.

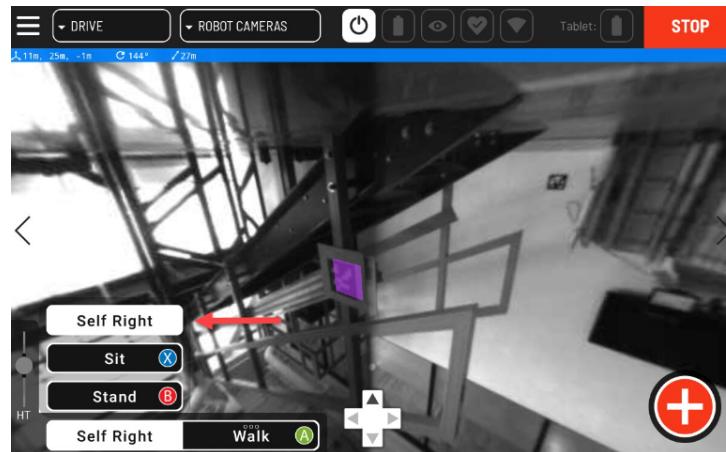


Imagen 16: Pantalla interactiva de la tablet

Scout

Por otro lado, se encuentra el software remoto Scout, que es una aplicación que permite a los operadores controlar los Spot desde una sala de control virtual o incluso desde una oficina en casa. Con Scout, se puede controlar al robot para trasladarlo a cualquier lugar (teleoperación).

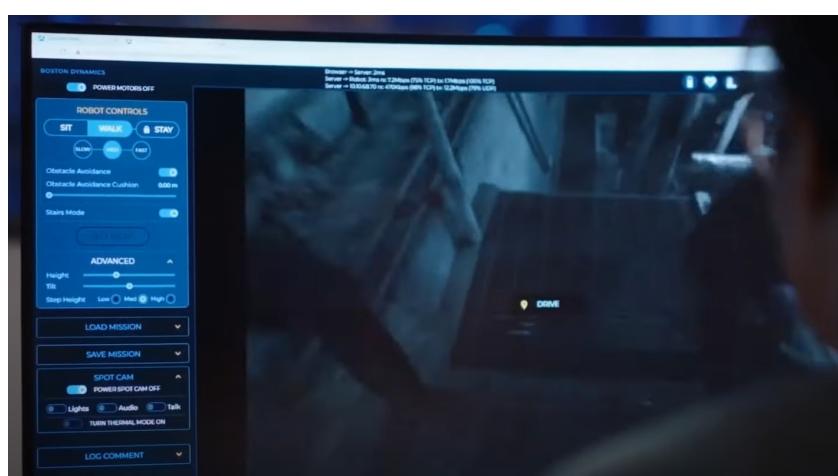


Imagen 17: Software remoto Scout

También se pueden ejecutar las misiones preprogramadas y, en modelos con dispositivos adicionales como Spot CAM+IR, que cuenta con cámaras esféricas, los operadores pueden capturar imágenes, datos térmicos y audio en tiempo real. Puede parecer que su utilidad es similar al de la tablet, sin embargo, este programa tiene varias ventajas adicionales. Por ejemplo, admite múltiples Spots y múltiples usuarios en un solo servidor. Además, varios espectadores pueden visualizar el mismo Spot a la vez, de forma que, si un operador encuentra un problema en la instalación, puede enviarle un enlace a su supervisor con la visión de ese robot Spot para que acceda a ella y evalúe la situación.

Scout se ejecuta en Site Hub, un dispositivo de red montado en rack 1U, que se incluye con el robot. El Site Hub se instala in situ y se comunica con sus Spots mediante WiFi.

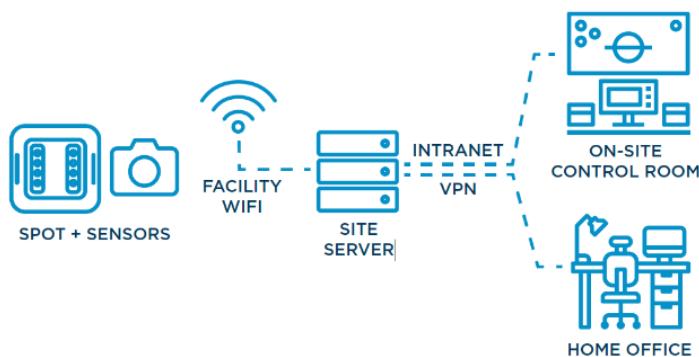


Imagen 18: Esquema de la conexión a Scout

Para facilitar su uso, el robot Spot incluye soluciones prediseñadas, APIs flexibles y SDKs programados en Python para crear controles personalizados y programar misiones autónomas.

Spot SDK

Debido a que Boston Dynamics es una empresa privada centrada en la investigación, existe una falta de información sobre algunos algoritmos que utiliza Spot. Para conocer más sobre la programación de este robot, por tanto, se ha estudiado en profundidad el repositorio de GitHub que contiene las diferentes funcionalidades SDK.

Entre todos los algoritmos utilizados para programar este robot, cabe destacar el uso de Tensorflow para la detección de objetos. Esta es una biblioteca de código abierto de Google para computación numérica y Machine learning a gran escala que viene pre-entrenada, permitiendo así la selección del objeto que se quiere detectar. Spot utiliza estos modelos pre-entrenados para ser capaz de detectar objetos. En la SDK se pueden encontrar diferentes script para utilizar esta funcionalidad como spot_tensorflow_detector.py, tensorflow_object_detection.py o spot_detect_and_follow.py. Para hacer posible esta detección se hacen uso de las 5 cámaras para capturar la imagen del objeto.

Spot también cuenta con un sistema de mapeo, localización y recorrido autónomo conocido como GraphNav, que consiste en un servicio que se ejecuta en el robot

Por otro lado, están los mensajes protobuf de Robot State, que guardan información sobre el robot.

La siguiente imagen muestra los datos devueltos al utilizar el servicio Robot State (RobotStateService) usando el ejemplo “get_robot_state” que, en este caso, pedía información de la articulación de cadera eje X de la pierna izquierda del robot.

```
...
joint_states {
    name: "f1.hx"
    position {
        value: 0.21174180507659912
    }
    velocity {
        value: 0.003905495163053274
    }
    acceleration {
        value: 2.1059951782226562
    }
    load {
        value: -1.86274254322052
    }
}
...
```

Imagen 19: Información sobre el FL.HX del robot

Para hacer funcionar este programa, hay que instalar el bosdyn API y cliente y ejecutarlo con python3. Las dependencias se pueden instalar usando el comando:

```
python3 -m pip install -r requirements.txt
```

Y para ejecutar el programa:

```
python3 get_robot_state.py ROBOT_IP {state, hardware, metrics}
```

Donde se debe indicar de cuál de los tres se quiere pedir información.

Por tanto, se puede utilizar para obtener información del servicio Robot State, que permite acceder a la configuración del hardware (esqueleto del robot y urdf), el estado del robot actual (Robot State) y las métricas del robot (métricas y parámetros).

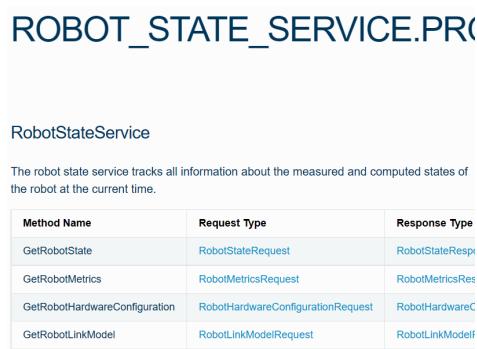


Imagen 20: Protobuf Robot State Service

Por otra parte, el protobuf Robot State también guarda información del estado de la batería, información y estados de errores, estados de comunicaciones como el Wifi, estados de E-Stop, información de la posición y el contacto de los pies del robot con el suelo, el estado del suelo (coeficiente de fricción estimada, normal respecto al suelo, deslizamientos, penetración del pie en el suelo...), estado del brazo manipulador (en caso de tenerlo), estado cinemático del robot (posición, velocidad...), etc.

ROBOT_STATE.PROTO

BatteryState

The battery state for the robot. This includes information about the charge or the temperature.

Field	Type	Description
timestamp	google.protobuf.Timestamp	Robot clock timestamp corr
identifier	string	An identifier for this battery
charge_percentage	google.protobuf.DoubleValue	Number from 0 (empty) to 1
estimated_runtime	google.protobuf.Duration	An estimate of remaining ru
current	google.protobuf.DoubleValue	Measured current into (char
voltage	google.protobuf.DoubleValue	Measured voltage of the en
temperatures	double	Measured temperature mea
status	BatteryState.Status	Current state of the battery.

BehaviorFault

The details of what the behavior fault consists of, and the id for the fault. The

Imagen 21: Protobuf Robot State

Existen muchos otros protobuf, como Spot_Cam/Led.Proto, que se encarga de analizar el estado y monitorizar o controlar el brillo de todos los LEDs o de uno en concreto; o Spot_Cam/Audio.Proto, que puede borrar, añadir o ejecutar un sonido de su librería, comprobar o cambiar el volumen, ver una lista con todos los sonidos del robot, etc; o incluso Spot/Door.Proto, que sirve para, por ejemplo, acercarse a una puerta y coger el pomo automáticamente o abrir puertas que solo requieren empujar, girar el pomo, ver el estado del Spot respecto a la puerta (abriendo puerta, puerta terminada de abrir...), etc.

SPOT/DOOR.PROTO

DoorCommand

Door Command specific request and Feedback.

DoorCommand.AutoGraspCommand

The robot searches along a ray for the door handle and automatically grasping door opening.

Field	Type	Description
frame_name	string	The name
search_ray_start_in_frame	bosdyn.api.Vec3	The start o
search_ray_end_in_frame	bosdyn.api.Vec3	The end o
hinge_side	DoorCommand.HingeSide	The side r

Imagen 22: Protobuf Spot/Door

Del mismo modo, existen muchos servicios que facilitan el acceso a estos protobuf. Sin embargo, se pueden dividir en siete servicios fundamentales:

- Robot State Service
- Image Service
- Local Grid Service
- World Object Service
- E-Stop Service
- Power Service
- Robot Command Service

El servicio Robot-State se ha visto anteriormente y guarda información del robot en tiempo real.

El servicio Image permite visualizar y controlar las diferentes cámaras e imágenes del robot (imágenes rgb y de profundidad).

El servicio Local-Grid se encarga de generar varios mapas de rejillas con información que recoge del entorno con sus sensores. Cabe destacar que este servicio no guarda o actualiza la rejilla en el tiempo, sino que crea una nueva en cada instancia de tiempo. La rejilla “terrain”, por ejemplo, guarda información de la altura estimada del terreno, mientras que la rejilla “terrain_valid” filtra valores de “terrain” que considera incorrectos (valores muy altos). “Intensity”, por su parte, guarda valores de color detectados en el terreno.

Por otro lado, los objetos muy altos detectados se consideran obstáculos y sus posiciones estimadas se guardan en “obstacle_distance”, donde el valor de cada celda indica la distancia al obstáculo más cercano (valores positivos a las afueras del obstáculo y negativos en el interior del mismo).

0	0	0	0	0	0	1	2	3	4	5
0	-1	-1	-1	0	1	2	3	4	5	
0	-1	-2	-1	0	1	2	3	4	5	
0	-1	-1	-1	0	1	2	3	4	5	
1	0	0	0	0	1	2	3	4	5	
2	1	1	1	1	1	2	3	4	5	
3	2	2	2	2	2	2	3	4	5	
4	3	3	3	3	3	3	3	4	5	
5	4	4	4	4	4	4	4	4	5	
6	5	5	5	5	5	5	5	5	5	

Imagen 23: Mapa de rejillas de obstacle_distance

Por último, la rejilla “no_step” guarda valores booleanos en cada celda según el robot determine que puede acceder a esa posición o no.

El servicio World-Object permite monitorizar y guardar objetos detectados en el mundo, asignándoles un nombre, un id, el tiempo que ha pasado desde la última vez que se ha visto, etc. También permite al usuario avisar al Spot sobre otros objetos que podrían detectarse.

El servicio E-Stop ofrece un mecanismo secundario de mantenimiento activo actuando como una parada de software. En su configuración, se especifica los endpoints esperados y su tiempo de espera. Durante la ejecución, se comprueban periódicamente todos los endpoints de forma que, si algún endpoint no se registra (o se registra pero niega la autoridad para operarlo), se apagará el motor del robot.

El servicio Power permite al cliente apagar y encender los motores del Spot.

El servicio Robot-Command, permite al cliente mover el robot y cuenta con muchos comandos pre-programados como “sentarse”, “ponerse de pie”, “moverse a posición x”, “moverse con velocidad y”, “parar”, “apagarse”, etc, para facilitar al usuario su control.

Simulación

Para comprobar el funcionamiento del robot de una manera más práctica se han buscado simulaciones de este robot en Gazebo que utilicen Ros.

Un ejemplo parecido e interesante es el SpotmicroAI versión customizada. Este es una versión más pequeña que el Spot de boston dynamic creado por Deok-yeon Kim y subido a Thingiverse. Actualmente tiene una licencia Creative Commons Attribution, donde cualquier persona es capaz de usar los códigos disponibles actuales además de crear nuevos y mejorar el proyecto.



Imagen 24: SpotmicroAI

En cuanto a la programación, este robot usa entrenamiento por refuerzo para calcular las constantes de control de movimiento y adaptarse a cualquier terreno. Para operar con él se puede hacer uso de un Joystick.

Para probar su funcionamiento se ha usado The Construct para crear un entorno de Ubuntu 18.04 con Ros Melodic y se ha descargado el repositorio de GitHub que contiene los paquetes de la simulación.

<https://gitlab.com/public-open-source/spotmicroai/simulation/-/tree/master/UserJo%C3%AB%20Martinez>

Estos paquetes tienen que situarse en el siguiente directorio `catkin_ws/src` y después compilar el espacio de trabajo.

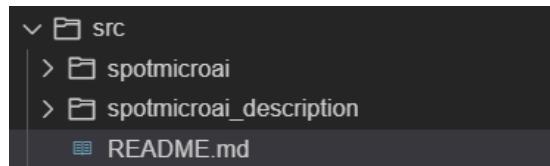


Imagen 25: paquetes `catkin_ws/src`

Para utilizar la simulación se hace uso del siguiente launch que abrirá gazebo cargando el robot y lanzando todos los nodos para su control. Estos nodos se pueden observar si hacemos `rostopic list`:

```
roslaunch spotmicroai_description SpotMicroAi.launch
```

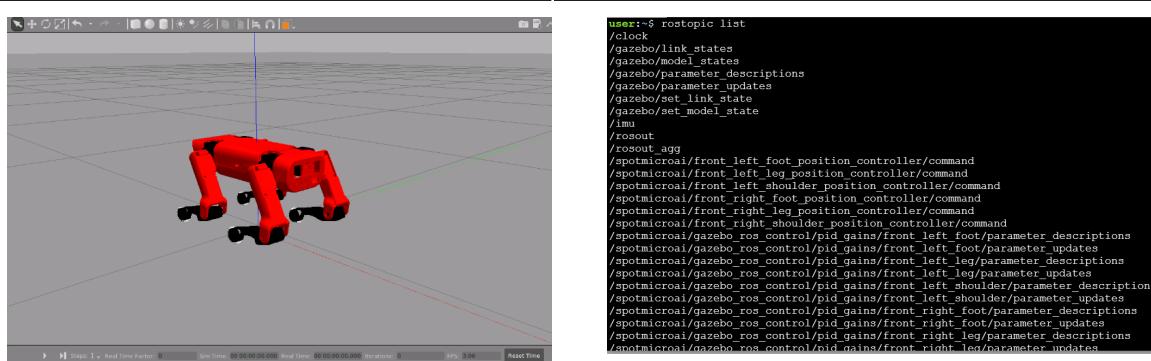


Imagen 26: Lista de nodos y simulación del Spot

El paquete dispone de otro .launch para realizar el control por Joystick, pero debido a que no disponemos de este, comentamos en el fichero el nodo que lo lanza y nos comunicamos directamente usando una terminal para publicar en el topic que utiliza este nodo.

```
roslaunch spotmicroai spotmicroai.launch
```

```

<launch>

    <!-- launch rosserial for joystick -->
    <node pkg="rosserial_python" name="serial_node" type="serial_node.py" >
        <param name="port" value="/dev/ttyACM0"/>
    </node-->

    <!-- launch legs nodes -->
    <node pkg="spotmicroai" name="Leg1" type="Leg.py" output="screen" args="_name:=Leg_1 _number:=0" />
    <node pkg="spotmicroai" name="Leg2" type="Leg.py" output="screen" args="_name:=Leg_4 _number:=1" />
    <node pkg="spotmicroai" name="Leg3" type="Leg.py" output="screen" args="_name:=Leg_2 _number:=2" />
    <node pkg="spotmicroai" name="Leg4" type="Leg.py" output="screen" args="_name:=Leg_3 _number:=3" />

```

Imagen 27: Fichero launch spotmicroai.launch

En el script Sit.py comprobamos que el topic que utiliza este para determinar el movimiento y en el que se enviaría el movimiento del Joystick se llama así JoyStick.

```

rospy.Subscriber("JoyStick", String, JoyStick_callback)
rospy.sleep(1)
while not rospy.is_shutdown():

    if key!=last and key !="stop":
        print(key)
        if key=="right":
            Legs=[0,0,1,1]
            movimiento.Move(Legs,Posiciones.SitBack)
            Legs=[1,1,0,0]
            movimiento.Move(Legs,Posiciones.SitFront)
        elif key=="up":
            Legs=[1,1,1,1]
            movimiento.Move(Legs,Posiciones.Arriba)
        elif key=="down":
            Legs=[1,1,1,1]
            movimiento.Move(Legs,Posiciones.Agachado)
    last=key

```

Imagen 28: Script Sit.py

Por lo tanto para mover el robot en la simulación de gazebo se utilizan comandos como los siguientes:

```

rostopic pub /JoyStick std_msgs/String "data: 'up'"
rostopic pub /JoyStick std_msgs/String "data: 'down'"
rostopic pub /JoyStick std_msgs/String "data: 'right'"

```

Aunque teóricamente el uso de estos comandos deberían haber simulado el movimiento del robot, este no se produjo por lo tanto se decidió probar otra simulación del mismo robot. Esta sería la del Spot Mini Mini OpenAI Gym Environment, esta se puede consultar en el siguiente repositorio.

https://github.com/OpenQuadruped/spot_mini_mini#dependencies

En este caso también se cuenta con un entrenamiento por refuerzo para adaptarse a diferentes entornos y que además se puede simular. Otros algoritmos utilizados serían el uso de curvas de bézier de 12 puntos para hacer un control de impedancia en el movimiento de trote, este algoritmo es el utilizado por el modelo Cheetah de Boston Dynamic.

Para poner la simulación en funcionamiento se usan los siguientes comandos con el fin de descargar los paquetes y las dependencias.

```
git clone https://github.com/OpenQuadruped/spot_mini_mini.git  
pip instal gym  
pip instal pybullet
```

Para que funcione, hay que colocar los paquetes descargados en la siguiente carpeta catkin_ws/src. Una vez los paquetes se han descargado correctamente se hace catkin_make.

Para poner en funcionamiento la simulación esta vez con pushbullet y sin la necesidad de usar un JoyStick se usan los siguientes comandos.

```
cd catkin_ws/src/spot_bullet/src  
.env_tester.py
```

Este script simula el robot y permite variar todos los parámetros para comprobar los diferentes funcionamientos que puede tener el robot, como se puede observar en el video adjuntado en el drive bajo el nombre “Rosject Simulation”.

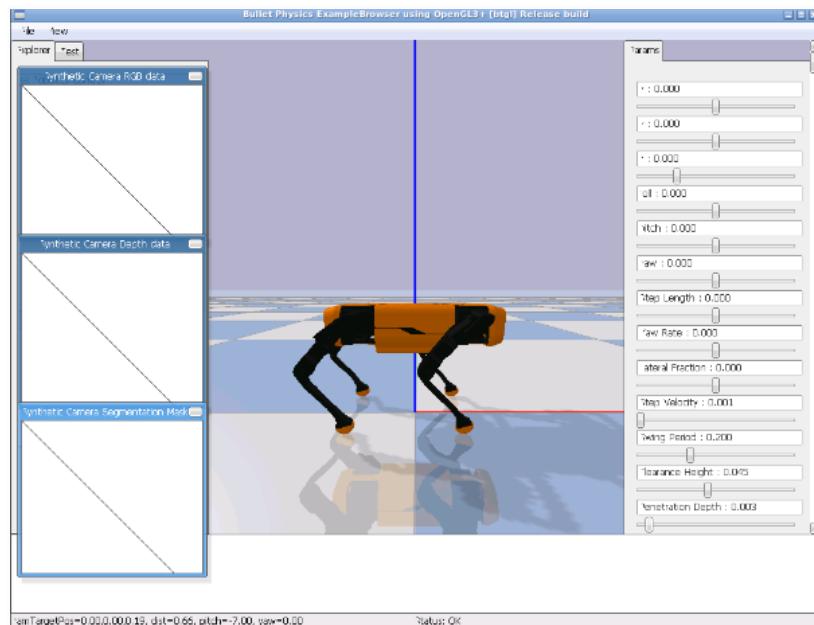


Imagen 29: Simulación Spot

Enlace al drive

A continuación, se muestra el enlace al drive donde se encuentra un video que resume el trabajo expuesto junto con el powerpoint utilizado para la explicación. También hay una carpeta con algunos de los documentos utilizados en la bibliografía:

<https://drive.google.com/drive/folders/1YhxSQhmZ6kNFHeAqdXaqUg1NSVGzcrA3?usp=sharing>

Bibliografía

[Boston_Dynamics#BigDog](#)

<https://www.bostondynamics.com/legacy>

<https://www.xataka.com/robotica-e-ia/wildcat-robot-cuadrupedo-de-boston-dynamics-capaz-de-correr-a-velocidades-de-25-km-h>

<http://wiki.ros.org/Robots/Spot>

<https://www.bostondynamics.com/products/spot>

[Spot® - The Agile Mobile Robot | Boston Dynamics](#)

[Doing More with Spot | Boston Dynamics](#)

[Spot \(bostondynamics.com\)](#)

[Spot ROS Computer Setup — Spot ROS User Documentation 0.0.0 documentation \(clearpathrobotics.com\)](#)

<https://www.bostondynamics.com/sites/default/files/inline-files/spot-specifications.pdf>

<https://github.com/boston-dynamics/spot-sdk>

<https://dev.bostondynamics.com/>

<https://support.bostondynamics.com/s/article/About-the-Spot-robot>

<https://aprendeia.com/que-es-tensorflow-como-funciona/>

https://github.com/boston-dynamics/spot-sdk/tree/master/python/examples/spot_tensorflow_detector

https://github.com/boston-dynamics/spot-sdk/blob/master/python/examples/spot_detect_and_follow/spot_detect_and_follow.py

[Scout application for Spot robot \(Boston Dynamics\) \(intuitive-robots.com\)](#)

[Scout.pdf \(intuitive-robots.com\)](#)

[About Spot — Spot 3.2.1.post1 documentation \(bostondynamics.com\)](#)

[Using the Robot State Service — Spot 3.2.1.post1 documentation \(bostondynamics.com\)](#)

[alerts.proto — Spot 3.2.1.post1 documentation \(bostondynamics.com\)](#)

[Robot Services — Spot 3.2.1.post1 documentation \(bostondynamics.com\)](#)

<https://spotmicroai.readthedocs.io/en/latest/>

<https://gitlab.com/public-open-source/spotmicroai/simulation>

<https://gitlab.com/public-open-source/spotmicroai/simulation/-/tree/master/UserJo%C3%ABl%C3%A9Martinez>

<https://dspace.mit.edu/handle/1721.1/98270>

[Autonomy — Spot 3.2.1.post1 documentation \(bostondynamics.com\)](#)

[Spot operational basics \(bostondynamics.com\)](#)

[What sensors are in Spot, the robotic dog? \(sensortips.com\)](#)